REDACTED VERSION OF DOCUMENT SOUGHT TO BE SEALED

KEKER & VAN NEST LLP
ROBERT A. VAN NEST - # 84065
rvannest@kvn.com
BRIAN L. FERRALL - # 160847
bferrall@kvn.com
DAVID SILBERT - # 173128
dsilbert@kvn.com
MICHAEL S. KWUN - # 198945
mkwun@kvn.com
633 Battery Street
San Francisco, CA 94111-1809
Telephone: 415 391 5400
Facsimile: 415 397 7188

Attorneys for Defendant ARISTA NETWORKS, INC.

UNITED STATES DISTRICT COURT

NORTHERN DISTRICT OF CALIFORNIA

SAN JOSE DIVISION

CISCO SYSTEMS, INC.,

Plaintiff,

v.

ARISTA NETWORKS, INC.,

Defendant.

Case No. 5:14-cv-05344-BLF (PSG)

ARISTA'S RESPONSIVE CLAIM CONSTRUCTION BRIEF

Tech Tutorial: March 11, 2016
Claim Construction Hearing: March 18, 2016

TABLE OF CONTENTS

TABLE OF AUTHORITIES

Page(s)

Federal Cases

Federal Statutes

I. INTRODUCTION

Cisco has proffered goal-oriented claim constructions that twist the patents to serve Cisco's litigation needs. For example, Cisco denies that "IOS"—a term that Cisco trademarked as the name of its own operating system—refers to Cisco's operating system, even though the specification makes clear that it does and a named inventor on that patent testified that it does. Likewise, Cisco denies that "XML" refers to the well-known and standardized XML language, even though that language was ubiquitous when Cisco drafted its patent, and Cisco has failed to cite a single contemporaneous document that used "XML" to mean anything else.

Arista's constructions, on the other hand, accord with the ordinary meaning of the claim terms and flow directly from the specification. As set forth more fully below, the Court should adopt Arista's proposed constructions.

II. LEGAL STANDARD

"[T]he interpretation to be given a [claim] term can only be determined and confirmed with a full understanding of what the inventors actually invented and intended to envelop with the claim." *Renishaw PLC v. Marposs Societa' per Azioni*, 158 F.3d 1243, 1250 (Fed. Cir. 1998). Claim construction should not "enlarge what is patented beyond what the inventor has described as the invention." *Netword, LLC v. Centraal Corp.*, 242 F.3d 1347, 1352 (Fed. Cir. 2001); *see also On Demand Mach. Corp. v. Ingram Indus., Inc.*, 442 F.3d 1331, 1340 (Fed. Cir. 2006) ("claims cannot be of broader scope than the invention that is set forth in the specification"). Instead, the correct claim construction "stays true to the claim language and most naturally aligns with the patent's description of the invention." *Phillips v. AWH Corp.*, 415 F.3d 1303, 1316 (Fed. Cir. 2005) (en banc).

III. '526 PATENT

A. Factual Overview

The '526 patent (Cisco's Br., Ex. 1) purports to make it easier for system administrators to utilize different management or diagnostic tools in complex computer systems. The patent explains that "system administrators may attempt to use multiple tools within a software system . . . for improved system performance." '526 patent, 1:28-31. A problem with doing so, however, is that this "requires the users to remember the names and syntaxes of numerous commands for the

respective . . . programs and . . . tools." *Id.* at 1:31-34.

The '526 patent purports to solve this problem by providing a "set of generic commands that are independent from the command formats" of the management programs, so that the user need only learn the single "generic" command set. *Id.* at 1:58-63. The "generic commands" are received by a parser (a piece of software), which translates them to "prescribed commands" that are unique to each management program. *Id.* at 1:48-63. According to the patent specification, this "arrangement provides the user a single set of commands and syntax to learn, facilitating the use of multiple administrative and maintenance tools." *Id.* at 4:58-60.

To translate the "generic commands," the parser uses a "command parse tree." *Id.* at 1:48-58. As shown in Figure 2 of the patent, a "command parse tree" is a tree structure with a root and hierarchically arranged nodes, or "elements," which each contain at least one "generic command component" (e.g., a word in a generic command) and a corresponding at least one "command action value," which is a value that the parser uses to identify a prescribed command. *Id.* at 1:48-54. The parser identifies the element of the tree that corresponds to the last valid generic command word to identify the corresponding prescribed command. *Id.* & Fig. 3.

B. Argument

1. "management programs" in Claims 1, 10, 14, and 23

Arista's Proposal	Cisco's Proposal
"tools that are configured to execute user-entered commands having their own respective command formats rather than the generic command format"	"separate tools or external agents having their own respective command formats that provide management functions"

The parties' constructions of this term differ in three ways: (1) Arista's proposal correctly reflects that management programs are configured to accept *user*-entered commands, not just machine-language commands; (2) Arista's proposal correctly reflects that a management program's "command format" must be different from the "generic command" format; and (3) Cisco's proposal improperly imports a "separate…or…external" limitation into the claims.

First, the Court should rule that "management programs" are configured to execute *user*-entered commands, as Arista's construction specifies. The parties agree that management

programs must have "their own respective command formats." Cisco, however, maintains that these command formats may consist of commands that only a machine (not a human) can issue.

Cisco's construction contradicts the '526 patent's purpose. The alleged invention purportedly eliminates the need for system administrators to learn and remember the many command names and syntaxes associated with multiple different tools. *See, e.g.,*'526 patent at 1:40-44 ("There is a need for an arrangement that integrates multiple [Real Time Monitoring] programs and command and control functionality for a user, ***without the necessity of learning the respective command formats and syntax***.") (emphasis added); *see also id.* at Abstract, 1:31-37, and 4:58-60. Under Cisco's construction, the invention would not achieve any particular benefit because Cisco would have the claim encompass systems using automated machine language instructions, which a systems administrator would never learn with or without the invention.

A claim construction that prevents an invention from achieving its stated purpose is likely incorrect. *Renishaw*, 158 F.3d at 1250 ("[T]he interpretation to be given a [claim] term can only be determined and confirmed with a full understanding of what the inventors actually invented and intended to envelop with the claim."). The Court should adopt Arista's construction that "management programs" are configured to execute *user*-entered commands.

Second, for similar reasons, the Court should rule that management programs' "respective command formats" must be different from the "generic command" format. Again, the alleged invention's purpose would be frustrated if the "generic commands" simply duplicated the original commands from the management programs. Presumably for this reason, the '526 patent and Cisco's own proposed construction for "generic command" recognize that generic commands use a "new syntax," "as opposed to the specific syntax for any such management program." '526 Patent at 3:30-35; *infra* § 2. Thus, the Court should rule that "management programs" use a command format that is different from the "generic command" format.

Third, the Court should reject Cisco's attempt to import a "separate . . . or external" limitation into the definition of "management programs." The portion of the specification that Cisco cites for this assertion actually refutes it:

> As shown in FIG. 1, the management programs 18, implemented ***for example*** by different OAM tools such as RTM programs, ***may***

> ***be executed within the processor based system or*** externally as external agents . . . . The management programs 18 ***may provide different administration and maintenance functions***, for example initiating various real-time screens used to monitor the internal state of executable processes within the software based system 10; ***alternatively, different tools*** 18 may allow the user to control the various states within the various component[s] of the software based system 10 via external programs (e.g., programs 18c or 18d), or may be used to issue external alarms . . . for external routines such as message waiting indicator routines.

'526 Patent at 3:1-15 (emphasis added).

As can be seen, the alleged support that Cisco relies on actually (1) states that management programs "may be executed within the processor based system," and thus *need not* be "external agents"; (2) never describes tools as "separate" (indeed, the word "separate" never appears in the patent at all); and (3) states that "different tools" are an *exemplary* embodiment of management programs, but not the only embodiment. *Id.* Rather, management programs "may provide different administration and maintenance functions," even if they are not "different tools." *Id.* In short, nothing in the specification says that management programs *must* be "separate tools" or "external agents."

Moreover, Cisco's invented "separateness" limitation is itself vague and in need of construction, and begs the question, "Separate from what?" The '526 patent provides no answer because it never talks about "separateness." Cisco states that a management program must be separate from "the same module or routine that is executed," (Cisco Br. at 3), but that answer provides no clarity, and in any event, Cisco has pulled it out of thin air. The ambiguity of Cisco's proposed construction is an independent reason to reject it. *See Chimie v. PPG Indus., Inc.,* 402 F.3d 1371, 1377 (Fed. Cir. 2005) ("Courts construe claim terms in order to assign a fixed, unambiguous, legally operative meaning to the claim.").

2. "generic command" in Claims 1, 6, 10, 13, 14, 15, 19, and 23

Arista's Proposal	Cisco's Proposal
Indefinite, OR if not indefinite: "command having a format and syntax that is an abstraction of the command formats and syntaxes of more than one management program, as opposed to the specific syntax for any such management	"command that provides an abstraction of the tool-specific command formats and syntax, enabling a user to issue the command based on the relative functions, as opposed to the specific syntax for a

program"	corresponding tool"

The '526 patent's purported point of novelty—the use of a "generic command" set—is so poorly defined that it is legally indefinite. No one—including the '526 patent's lead inventor—knows with reasonable certainty what is, or is not, a "generic command."

A patent is invalid for indefiniteness under 35 U.S.C. § 112 if "its claims, read in light of the specification . . . and prosecution history, fail to inform, with reasonable certainty, those skilled in the art about the scope of the invention." *Nautilus, Inc. v. Biosig Instruments, Inc.*, 134 S. Ct. 2120, 2123 (2014). The purpose of this "definiteness requirement" is "to ensure that the claims delineate the scope of the invention using language that adequately notifies the public of the patentee's right to exclude." *Datamize, LLC v. Plumtree Software, Inc.*, 417 F.3d 1342, 1347 (Fed. Cir. 2005). "Otherwise, competitors cannot avoid infringement, defeating the public notice function of patent claims." *Halliburton Energy Services, Inc. v. M-I LLC*, 514 F.3d 1244, 1249 (Fed. Cir. 2008).

Indeed, to uphold claims like these, which do not clearly define the scope of the invention, would thwart the ultimate purpose of the patent laws: to "promote the Progress of Science and useful Arts." *Festo Corp. v. Shoketsu Kinzoku Kogyo Kabushiki Co.*, 535 U.S. 722, 730 (2002) (quoting U.S. Const. art. I, § 8, cl. 8). A patent claim's boundaries should be clear. This clarity is essential to promote progress, because it enables efficient investment in innovation. A patent holder should know what he owns, and the public should know what he does not." *Id.* As the Supreme Court held over 80 years ago:

> To sustain claims so indefinite as not to give the notice required by the statute would be in direct contravention of the public interest which Congress therein recognized and sought to protect.… The statutory requirement of particularity and distinctness in claims is met only when they clearly distinguish what is claimed from what went before in the art and clearly circumscribe what is foreclosed from future enterprise. A zone of uncertainty which enterprise and experimentation may enter only at the risk of infringement claims would discourage invention only a little less than unequivocal foreclosure of the field.

United Carbon Co. v. Binney & Smith Co., 317 U.S. 228, 233 (1942).

The '526 patent's claim term "generic command" is indefinite because one of skill in the art cannot determine with reasonable certainty, or any certainty, what is or is not "generic." The

term "generic command" is not a term of art with a well-known meeting outside the context of the '526 patent.

Arista's proffered expert, Dr. Kevin Almeroth, confirmed that the term is indefinite. He stated that "generic commands" provide an "abstraction" of the tool-specific command formats and syntax, and that "[i]n Computer Science, abstraction is a concept that allows suppressing the specific details of how something is implemented . . . ." (Dkt. 91-1, Almeroth Decl., ¶ 69.)

This ambiguity is fatal. Just as the '526 patent's lead inventor and Cisco's own expert cannot determine with reasonable certainty whether a particular command is or is not a "generic

command," other skilled artisans cannot do so either. As a result, the '526 patent fails to "clearly distinguish what is claimed" and "clearly circumscribe what is foreclosed from future enterprise," creating a "zone of uncertainty which enterprise and experimentation may enter only at the risk of infringement claims . . . ." *United Carbon,* 317 U.S. at 236. Because Cisco has failed to meet the statutory requirement to claim its alleged invention with distinctness and particularity, the Court should find the asserted claims of the '526 patent invalid as indefinite. *See Datamize*, 417 F.3d at 1350 (finding claim requiring an "aesthetically pleasing look and feel" indefinite); *Interval Licensing LLC v. AOL, Inc.*, 766 F.3d 1364, 1373 (Fed. Cir. 2014) (finding the claim term "unobtrusive manner" indefinite).

Alternatively, should the Court construe this term, Arista's construction is more appropriate. Both constructions derive from the same sentence in the specification. '526 Patent, 3:30-35. But Arista modified that sentence—which describes a generic command ***set***—because the claim language at issue is "generic command" (not "generic command set"). Cisco's unmodified version conflates "generic command" with "generic command set," thereby mixing apples and oranges.

3. "respective command formats," in Claims 1, 10, 14, 23

Arista's Proposal	Cisco's Proposal
"command names and syntaxes specific to each management program"	"command format specific to a management program"

Due to space limitations, Arista submits its argument on "respective command formats" based on the evidence cited in the Joint Claim Construction Statement.

4. "command parse tree," in Claims 1, 3, 10, 11, 12, 14-16

Arista's Proposal	Cisco's Proposal
"tree": "data structure consisting of linked nodes, with a root node (a node with no parent nodes), and where the remaining nodes are either a branch node (a node with a parent node and one or more children nodes), or a leaf node (a node with a parent node and no children nodes)" "command parse tree": "tree for interpreting commands where each node corresponds to a command component"	"a hierarchical data representation having elements each specifying at least one corresponding generic command component and a corresponding at least one command action value"

The principle difference between the parties' constructions is that Arista's construction properly defines a "tree" as a "data structure," while Cisco's broadens the term to include "data *representation[s]*." Cisco apparently hopes to sweep in software that is not a tree, but which Cisco will argue has some type of hierarchical characteristic. But in computer science, a "tree" is a data structure—as the technical dictionary definitions submitted by *both* parties, and the testimony of *Cisco's* expert, make clear. (Ex. 3, Microsoft Press Computer Dictionary (1997) at 477; Ex. 4, Barron's Dictionary of Computer and Internet Terms (1998) at 476-77; Ex. 5, Webster's New World Dictionary of Computer Terms (1999) at 536; Almeroth Decl., ¶ 73).

Similarly, whereas Cisco would describe a tree as merely "hierarchical," Arista's precise construction adopts the common computer science understanding—reflected in the patent—that a tree begins with a "root" node, and then fans out into branch and leaf nodes. *See* '526 Patent at Fig. 2 (showing a tree with a "Root" node, branch nodes, and leaf nodes). Indeed, all three cited dictionary definitions, Cisco's own brief, and Cisco's expert declaration reflect this understanding. (Cisco Br. at 10) ("A tree is a hierarchical structure comprised of sub-trees that can themselves be treated as trees."); (Almeroth Decl., ¶ 73) ("Each sub-tree has the same characteristics, namely a root node with child nodes"); (*id.* ¶ 74) (including multiple depictions of trees with root nodes linked to branch and leaf nodes).

Cisco also criticizes Arista's construction of "tree" on the ground that it precludes the possibility of a zero- or one-node tree. But a "tree" with zero nodes or one node would not meet Cisco's definition either, because it would not be "hierarchical." A "tree" with no nodes or only one node is an abstract, paradoxical concept (suggested nowhere in the '526 patent) that would only confuse the jury, and the Court should not allude to it in its claim construction.

The remainder of Arista's construction correctly explains how the '526 patent uses the term "command parse tree." The portion of Cisco's construction that follows "a hierarchical data representation"does not belong in the definition of "command parse tree" because it repeats verbatim claim limitations that follow "command parse tree."

5. "the validating step including identifying one of the elements as a best match relative to the generic command" in Claims 1 and 14, and the closely related terms in Claims 10 and 23

Arista's Proposal	Cisco's Proposal
Indefinite[1], OR if not indefinite: "the validating step[2] having the capability of both identifying the element in the parse tree that exactly matches the generic command, and, in the absence of an exact match, identifying the element that contains the last validated component of the generic command"	Plain and ordinary meaning (except for specific terms appearing within the phrase that are otherwise terms for construction)

This dispute focuses on the meaning of "best match." Cisco's interpretation of the term's "plain and ordinary meaning" improperly reads the word "best" out of the claims. Cisco maintains that a parser that can only identify an exact match would nevertheless satisfy the "best match" limitation. (Cisco Br. at 9-10). If that were correct, the claims would simply cover "a match relative to the generic command," not "a *best* match." The proper construction must give meaning to all of the words of the claim.

The patent's only disclosure of its "matching" functionality comports with Arista's construction. It contains two examples of validating a "generic command." In the first example, "the parser … receives the ***valid*** command 'watch tcp connections.'" '526 Patent, 4:7-37 (emphasis added). In this example, each word of a ***valid*** command is matched to an element of the command parse tree, such that when the parser reaches the last word or token—*i.e.*, the word "connections"—the parser "identif[ies] a match between the entire generic command" and the correct "token-command key" pair (which is found in box 24c of Figure 2). *Id.* at 4:27-31 & fig. 2. (Notably, the independent claims use the phraseology "generic command component" and "command action value," respectively, instead of "token" and "command key"—but the concepts are the same.) In this example, the parser identifies an exact match.

[1] The claim term is indefinite because it contains the phrase "generic command." *See* Section III.B.2, *supra*.

[2] Claims 10 and 23 have an identical requirement, except that they refer to a "parser" or "validating means," rather than a "validating step." Thus, for those claims, the "parser" or the "validating means" would have the proposed capability.

In the second example, the exemplary generic command is "get udp connection info." While "get" is a valid first word, "udp" is not a valid word to follow "get." *Id.*at 4:39-46. As such, when the parser reaches the invalid word "udp," "the parser **14** uses the last valid command key…based on the matching token for the first valid word located [*i.e.*, the command key for the word "get"]." *Id.* at 4:46-48. In other words, the specification explains that, in the case of a ***partially valid command***, the parser matches to the "command key" (or, in the terminology of the claims, "command action value") of the last valid word. *See also id.* at Fig. 3 (illustrating the algorithm for "matching" either valid or a partially valid command). The parser therefore identifies an exact match if one exists, and if none exists, it identifies an alternative match by matching the last valid command word. This is the only disclosure of anything resembling a "best match" feature.

Although the disclosures in the specification are sparse, they are the only support for the "best match" claim language. Accordingly, the Court should adopt Arista's construction, which honors the presence of the word "best" in the claims.

6. "recursively traversing the command parse tree based on an order of the input command words" in Claims 3 and 16

Arista's Proposal	Cisco's Proposal
"recursively": "by using a function that calls itself," "traversing the command parse tree based on an order of the input command words": "sequentially determining the presence of each word of an input command in a node of a command parse tree, such that the order of the words (e.g., first, second, third) corresponds to the hierarchy of the nodes (e.g., parent, child, grandchild)."	Plain and ordinary meaning (except that specific terms appearing within the phrase should be construed as proposed above)

Arista's proposed construction gives the term "recursive" its ordinary meaning in computer science, namely a function that calls itself. Cisco, by contrast, has offered no proposed construction for "recursive"—a solution that is plainly unworkable, because jurors will not know what "recursive" means.

There is no meaningful intrinsic evidence defining the term "recursively." The one use of "recursively" in the specification provides no guidance. *See* '526 Patent, 3:54-57. Arista provides

four contemporaneous technical dictionaries for the simple proposition that in computer science, "recursion" refers to a function that calls itself. (Ex. 3, Microsoft Dictionary at 400) ("The ability of a routine to call itself."); (Ex. 4, Barron's at 389) ("the calling of a procedure by itself, creating a new copy of the procedure"); (Ex. 5, Webster's at 451) ("In programming, a program instruction that causes a module or subroutine to call itself."); (Ex. 11, Random House Webster's Computing & Internet Dictionary 3rd Ed (1999) at 471) ("A programming method in which a routine calls itself.").

By contrast, no contemporaneous extrinsic source supports Cisco's expert's assertion that "recursively traversing" means accessing all the nodes of a tree in a particular order. Dr. Almeroth cites the Microsoft dictionary entry for "traversing" (*i.e.*, to access in a particular order all of the nodes of a tree), but that source says nothing about what it means to traverse a tree ***recursively***, as the claim language requires.

7. "the command parse tree having elements each specifying at least one corresponding generic command component and a corresponding at least one command action value" in Claims 1 and 14

Arista's Proposal	Cisco's Proposal
"elements": "nodes" "command action value": "piece of data that uniquely represents the prescribed command." the entire phrase: "the command parse tree having nodes, such that each node specifies a unique command action value for each generic command component."	Plain and ordinary meaning (except for specific terms appearing within the phrase that are otherwise terms for construction)

This phrase requires construction because it appears necessary to clarify that, in the command parse tree, there must be a one-to-one pairing between "generic command components" and "command action values." The claim language permits an element (or node) of the tree to have one or more generic command components. But the phrase "a corresponding at least one command action value" means that the number of command action values must correspond to the number of generic command components. This is exactly what the patent teaches. *See* '526 Patent, Fig. 2 (showing each "generic command component" (depicted as tokens, designated as "T") is paired with exactly one command action value (depicted as command keys, designated as

"CK")).

Cisco attempts to obscure the issue by observing that in Appendix A of the patent, there is a "new syntax" command "watch acb globals" that is associated with two previous commands, "BASEview" and "APPview." [redacted].

Finally, Cisco complains that Arista's construction requires a command action value to be a piece of data. A tree is a data structure. *See supra* § 4. And per the claim language, a command action value resides in the elements of the command parse tree. *See also* '526 Patent, Fig. 2 (showing the command action value, depicted as command keys, in the nodes of the command parse tree). It therefore follows that a command action value is a piece of data. Moreover, this accords with the specification; the only command action values depicted in the patent are numbers between 1 and 13. *See id.* (showing "CK" values between 1 and 13).

Cisco's proposal, on the other hand, is unhelpful. The phrase "command action value" has no plain and ordinary meaning outside the '526 patent, and jurors will not understand it.

8. "prescribed command of a selected one of the management programs" in Claims 1 and 14

Arista's Proposal	Cisco's Proposal
"one of the commands in the respective command format of a specific management program"	Plain and ordinary meaning (except for specific terms appearing within the phrase that are otherwise terms for construction)

Cisco's sole complaint about Arista's construction is directly contradicted by the claim language. Cisco objects that Arista "limit[s] the prescribed command to only a single, specific

management program." (Cisco Br. at 13). But the claim language provides that a prescribed command is "of a selected ***one*** of the management programs" (emphasis added). And because the prescribed command is "of" a particular management program, it must be in the format that that management program would understand. This is precisely the purpose of the purported invention: the parser keeps track of the various command formats of the various management programs so that the user need not do so. *See, e.g.*, '526 Patent, 1:41-44.

9. command word translation table, configured for storing for each prescribed command word a corresponding token" in Claims 2 and 15

Arista's Proposal	Cisco's Proposal
"command word translation table": "data structure with rows and columns that translates a command word into a token" "prescribed command word": "valid generic command word" "token": "letter, number, ~~or~~ symbol, [**or other short code**] that either (1) uniquely represents a prescribed command word, or (2) indicates the presence of an invalid command word" The remainder of the phrase does not require construction	Plain and ordinary meaning (except for specific terms appearing within the phrase that are otherwise terms for construction)

Arista's proposal that a "table" is a "data structure with rows and columns" is supported by both the specification (*see* fig. 2, item 20) and by four technical dictionaries. (Ex. 3, Microsoft at 459); (Ex. 4, Barron's at 456); (Ex. 10, IBM Dictionary of Computing (1994) at 679, 704) (defining both table and translation table); (Ex. 11, Random House Webster's at 544). Cisco's attorney argument that a table can be anything that stores "for each prescribed command word a corresponding token" is unsupported by any evidence, and would improperly read the word "table" out of the claim.

With respect to the term "prescribed command word," this phrase appears only once in the specification, where it refers to the left-hand column of translation table 20 in Figure 2. '526 Patent, 3:39-43. That column is populated with the possible valid words for an input generic command—which is precisely what Arista's construction captures. Cisco provides no explanation of what else the term could mean.

Finally, Cisco criticizes Arista's construction of the term "token"—a term with a

specialized meaning in the '526 patent—but seeks to leave the term undefined. Tokens are clearly presented in the translation table and command parse tree of Figure 2. The tokens' only apparent purpose is saving space: they avoid the need for the system designer to repeatedly insert the entire prescribed command word into the command parse tree. For this type of token to have any value, it must be significantly shorter than the prescribed command word that it represents (most of which are only three letters long). To address Cisco's criticism that a token could be "more than one letter, number, or symbol," Arista has modified its construction to reflect that a token can also be some other "short code" that represents a prescribed command word. With this modification, Arista's construction should be adopted.

10. "means for validating a generic command received from a user, the validating means configured for specifying valid generic commands relative to a prescribed generic command format and having elements each specifying at least one corresponding generic command component and a corresponding at least one command action value, the validating means identifying one of the elements as a best match relative to the generic command" in Claim 23

Arista's Proposal	Cisco's Proposal
Functions: (1) validating a generic command received from a user (2) specifying valid generic commands relative to a prescribed generic command format, (3) having elements each specifying at least one corresponding generic component and a corresponding at least one command action value, and (4) identifying one of the elements as a best match relative to the generic command. Disclosed structure: A processor executing a parser, and a corresponding memory storing a command parse tree, wherein the parser executes the algorithm of Figure 3, and wherein (1) each node of the command parse tree specifies one token and a corresponding command key; (2) the top-level nodes of the command parse tree represent all possible valid first words in the input command, second-level nodes represent all possible valid second words for each valid first word in the input command, and so on;	For "means for validating a generic command received from a user": Function: validating a generic command received from a user. Structure: Parser 14 in Figure 2, which includes the command word translation table 20 and the command parse tree 22, as described in 3:36- 61, and equivalents For rest of term, plain and ordinary meaning (except that specific terms appearing within the phrase should be construed as proposed above)

Arista's construction complies with the law governing means-plus-function claims by

identifying every function stated in the claim, and by identifying the corresponding structure in the specification for each such function. Without citing any law, Cisco contends that it may ignore some of the stated functions by labeling them "sub-functions" (Cisco Br. at 15).

Paragraph 6 of (pre-AIA) 35 U.S.C. § 112 governs means-plus-function claims. When a claim limitation uses the term "means" it is presumed to invoke § 112, paragraph 6, unless "the claim language specifies the exact structure that performs the function in question without need to resort to other portions of the specification or extrinsic evidence for an adequate understanding of the structure." *TriMed, Inc. v. Stryker Corp.*, 514 F.3d 1256, 1259-60 (Fed. Cir. 2008); *Altiris, Inc. v. Symantec Corp.*, 318 F.3d 1363, 1376 (Fed. Cir. 2003) (holding that even though the claim language "means of booting" specified implementation with "commands for booting," and even though "commands" are structural, there was insufficient structure to overcome the presumption, because "commands for booting" is "really just a restatement of 'means of booting'").

Here, the three purported "sub-functions" are all phrased functionally: "specifying," "having elements each specifying," and "identifying" are functions. Cisco recognizes as much by calling them "sub-functions." Accordingly, all three sub-functions, and their corresponding structures, must be included in the construction.

Cisco's remaining arguments ignore the specification. First, Cisco appears to argue that the corresponding structure need only be able to handle valid commands. But the claim language requires identification of a "best match," which—as explained above—can happen only if the parser is able to issue a prescribed command even when the entered command is only partially valid. *See* Section III.B.5, *supra*.

Second, Cisco argues that the specification does not require paragraph (2) of Arista's construction (*i.e.*, that the top-level nodes of the command tree represent all possible valid first word, and so on); but that is ***exactly*** what the specification requires. Cisco cannot seriously dispute that the top-level node in Figure 2 (*i.e.*, item 24a) represents possible first words, that the second-level nodes (*i.e.*, items 24b, 24e, and 24h) represent possible second words, and so on. *See, e.g.*, 4:5-31 (for the valid command "watch tcp connections," finding a match for "watch" in the first level node 24a, finding a match for "tcp" in the second level node, and finding a match

for "connections" in the third-level node 24c). Moreover, if the command parse tree did not contain ***all possible*** valid word combinations, the parser would not be able to recognize those purportedly valid commands, and the purported invention would not work as to those commands.

Cisco's proposal—that the disclosed structure is limited to the supposed "embodiment" disclosed in column 3, from lines 36 to 61—makes no sense, because Cisco seeks to exclude the corresponding disclosure (*see* 3:62-4:54 & Fig. 3) that explains how Cisco's cited embodiment works. Figures 2 and 3 are obviously complimentary disclosures of the same embodiment: Figure 2 depicts the relevant data structures (the translation table and parse tree) while Figure 3 depicts the decisional logic that applies to these data structures. This is apparent from the fact that the entire verbal description of Figure 3 continuously refers to Figure 2. '526 patent, 4:3-54. Cisco's proposed structure is therefore incomplete.

IV. '886 PATENT

A. Factual Overview

The '886 patent (Cisco's Br., Ex. 2) sought to provide an easier, more structured approach to sending input to—and receiving output from—Cisco routers. '886 Patent, 1:26-38. The disclosed invention is a "parser subsystem" that receives input command data broken down in a structured format. In all of the embodiments, as well as all of the claims, that structured format is XML. The subsystem's parser translates the XML into a CLI command by traversing the XML fragments of the input according to the rules of the CLI syntax. After the translation from XML to CLI, the subsystem passes the command on to the main IOS/CLI system.

In addition to receiving commands from outside the system, the parser subsystem also receives CLI statements from the IOS/CLI system. The subsystem translates these CLI statements into XML by parsing the statements into "tokens," and then applying XML tags around the individually parsed elements. The translated XML statements are sent outside the system to the user.

B. Argument

1. "extensible markup language (XML)" in Claims 1-10

Arista's Proposal	Cisco's Proposal

"markup language defined by one of the versions of the XML standard published by the W3C organization"	"extensible": a property of a computer language that allows the user to add new features or modify existing ones "markup language": a computer language that allows the user to add identifiers to a document for indicating logical components or layout

There is no dispute that XML (as it is used exclusively in the specification), or "extensible markup language (XML)" (as it is used in the claims), has a well-understood specific meaning that is defined by the W3C, also known as the "World-Wide Web Consortium." (Ex. 6, Webster's New World Computer Dictionary (2001) at 399; Ex. 7, Microsoft Computer Dictionary (2002) at 578; Ex. 8, Barron's Dictionary of Computer and Internet Terms (2003) at 558-60). Although Cisco endorses the principles that "the words of a claim are generally given their ordinary and customary meaning," and that "[t]echnical dictionaries are particularly useful in understanding the ordinary and customary meaning of claim terms," Cisco's Br. at 16 (citing *Philips*, 415 F.3d at 1312-13, 1318), Cisco promptly ignores those rules to support a construction of XML that only litigation counsel could conceive.

Like Hypertext Markup Language (HTML)—which was also standardized by the W3C—XML was a particular, defined language at the time of the patent. Contrary to Cisco's suggestion, XML did not refer generically to any language that used markup and that was extensible. Indeed, Cisco cites no evidence—aside from the *ipse dixit* of its paid expert—to support the notion that "extensible markup language (XML)" was understood at the relevant time to mean something other than the W3C-standard language.

Cisco cobbles together a construction based on dictionary definitions of the words "extensible," "markup," and "markup language." But like many technical acronyms, "XML" has a meaning in the field that is more specific than the sum of the discrete words that make up the acronym. XML is a *name* given to a particular language; if it were simply a shorthand for three discrete words, one would expect it to be "EML." Moreover, the claims' use of "extensible markup language" does not expand its meaning beyond the commonly understood industry-standard acronym "XML" because the spelled-out phrase is always immediately followed by

"XML" within parentheses.

Cisco's parsing of the words of this commonly understood term is unprecedented. Under Cisco's rule, the term "Uniform Resource Locator (URL)"—which refers to a standardized naming convention for addressing documents accessible over the Internet—would simply mean any "uniform" way of "locating" a "resource," and not the format used billions of times a day by Internet users. The absurdity of Cisco's construction is plain.

In the specification of the '886 Patent, the use of "XML" comports with its customary technical meaning. For example, the XML schema in Table 1 is W3C-standard XML. (*See* Ex. 9, Tjong Depo at 46:3-24.) Cisco charges that Arista's reference to this example is an effort to limit the asserted claims to a particular embodiment. Not so. It is Cisco, not Arista, who is urging the Court to adopt a non-standard construction of "extensible markup language (XML)." Cisco, therefore, must show that the patentee clearly altered or disclaimed the term's ordinary meaning. *See Ancora Techs., Inc. v. Apple, Inc.*, 744 F.3d 732, 734 (Fed. Cir. 2014) *cert. denied*, 135 S. Ct. 957 (2015) ("A claim term should be given its ordinary meaning in the pertinent context, unless the patentee has made clear its adoption of a different definition or otherwise disclaimed that meaning.").

Cisco points to two instances in the patent that purportedly support Cisco's construction, but neither comes close to showing a clear adoption of Cisco's creative definition. ***First,*** Cisco stresses that the claims recite "*an* extensible markup language (XML) format," Cisco's Br. at 18 (emphasis in original), thus suggesting that the indefinite article "an" suggests multiple possible "extensible markup language[s] (XML)." But the proper reading is that "an" suggests multiple possible "*formats*," not XMLs. For example, the reader of a want ad for "a Super Bowl ticket" would assume that the indefinite article "a" modifies "ticket," not "Super Bowl." Because "Super Bowl" is commonly understood to refer to the annual NFL championship game, the reader would not understand the term to refer to tickets to just any particularly exceptional bowl. While there may be tens of thousands of Super Bowl *tickets*, there is only one Super Bowl. Similarly, a reference to "a HyperText Markup Language (HTML) document" would indicate that there could be multiple HTML *documents*, not multiple examples of hypertext markup languages. The same

reasoning applies to "an extensible markup language (XML) format." While there certainly can be different XML formats, those formats are built using the standardized XML language. It is the "formats" that are indefinite, not the XML language itself.

Second, Cisco points out that the specification states that the invention is not limited to XML. While that may be true, the claim is plainly limited to XML and Cisco either was unable, or chose not, to obtain claims on non-XML embodiments. In any event, the cited portion does not support Cisco's position that "extensible markup language (XML)" does not in fact have the conventional meaning of "XML." On the contrary, the passage, which provides that the commands can be "*presented in a language other than XML or CLI*," 886 Patent, 3:26-29 (emphasis added), confirms that "XML" is a specific language familiar to those in the art, and not some special definition. *See Ancora*, 744 F.3d at 734.

Because Arista's construction of "extensible markup language (XML)" is consistent with the understanding of the term at the time of the invention, and because Cisco fails to show that the patentee clearly used a non-standard definition, the Court should adopt Arista's construction.

2. "command line interface (CLI) parser" / "receiving, with a command line interface (CLI) parser" in Claims 1-5

Arista's Proposal	Cisco's Proposal
"receiving": "taking as an input" "a command line interface (CLI) parser": "a program that breaks down the individual sub-parts of a command using a CLI grammar"	"a component of the routing system for analyzing command line interface (CLI) commands using a grammar"

The parties' primary disagreement here centers on the meaning of the term "parser." Arista's construction—which defines a CLI parser as a program that breaks down input into "individual subparts"—is consistent with the common technical understanding of the term. *See, e.g.,* (Ex. 6, Webster's at 275: "**parser 1.** A program that breaks large units of data into smaller, more easily interpreted pieces."); (Ex. 7, Microsoft Dictionary at 392: "**parser** *n.* An application or device that breaks data into smaller chunks so that an application can act upon the information"). At the time of the invention, artisans in the field of computer science understood the term "parser" to refer to a program that breaks down data into smaller pieces or chunks.

Indeed, the extrinsic evidence cited by Cisco supports this understanding. In the Joint Claim Construction Statement, Cisco cites the following definition in support of the construction of this claim term:

> Parse: 1. To examine closely and break down into components. 2. In computer, to analyze ***and separate into components*** which are more easily processed, converted or the like.

(Dkt. 70-1, Cisco's Ex. A at 20) (citing *Wiley Electrical and Electronics Engineering Dictionary*) (emphasis added). In short, even the extrinsic evidence cited by Cisco supports the understanding that a parser must break down an input string into smaller chunks.

While Arista's construction clarifies the meaning of "parser" in light of the common understanding of the term, Cisco's construction obscures that meaning. In effect, Cisco's construction replaces "parsing"—which has a specific meaning in the art—with "analyzing," a more generic verb that would frustrate a jury's attempt to understand the bounds of the claims.

3. "internetwork operating system (IOS) command line interface (CLI) parser subsystem" in Claims 2-4, 7-9

Arista's Proposal	Cisco's Proposal
"internetwork operating system (IOS) command line interface (CLI)": "the CLI interface designated by Cisco as IOS CLI and that is used for configuring, monitoring, and maintaining Cisco devices. Also referred to as 'IOS/CLI.'" the entire phrase: "processor or portion thereof that executes a program that breaks down the individual sub-parts of a command using the grammar of IOS/CLI"	"a subcomponent of the internetwork operating system (IOS) for analyzing command line interface (CLI) commands using a grammar"

For this term, the central issue is whether the claim phrase "internetwork operating system (IOS) command line interface (CLI)" refers specifically to a Cisco interface. The rest of the term—which concerns the construction of "command line interface (CLI) parser"—is addressed in Section IV.B.2, *supra*.

[redacted] Cisco clearly understood that

"IOS CLI" referred to a Cisco product. In fact, Cisco applied for—and was issued—a trademark on the word "IOS" in 1998. *See* Registration No. 2,214,831.

The term is used the same way in the '886 patent. The patent places the problem to be solved in the specific context of Cisco's IOS CLI:

> IOS CLI is not the most program-friendly of interfaces, however. Twenty years of consistency and backwards-compatibility, when coupled with consistent improvements to the hardware and implementation of new features, has created an extensive interface. While a human user of IOS CLI may be able to sort through the complicated input and output scheme to input information and extract important data, it has proven to be a very difficult and cumbersome task to automate.

'886 Patent, 1:26-33. The specification here is clearly referring to a particular product, not to command line interfaces in general. And that product is Cisco's. No other company sold a command line interface that was known as "IOS CLI," let alone one that had been used for twenty years. When viewed in the context of the understanding of the term "IOS" at the time, the patent uses "internetwork operating system (IOS) command line interface (CLI)" to refer to Cisco's IOS CLI.

Cisco's answer is to point out that "Arista's proposed construction creates the absurd result that the only entity that can possibly infringe a claim in a Cisco-owned patent is Cisco itself." (Cisco's Br. at 21). But Cisco forgets that the full claim term is "internetwork operating system (IOS) command line interface (CLI) ***parser subsystem***." An entity other than Cisco could potentially use, make, or sell a subsystem that interfaced with Cisco's IOS CLI product. The patent itself acknowledges as much, explaining in the specification that "many consumers have invested heavily in IOS CLI support, developing complicated scripts to handle various configuration and access needs." '886 Patent, 1:20-23. According to the specification, the patentee was motivated, in part, by the efforts of others to provide improved access and configuration capabilities in connection with IOS CLI. *See id.* at 1:12–25. It was thus clear to Cisco at the time of the invention that non-Cisco entities could build subsystems that interfaced with Cisco's IOS CLI product.

In sum, both the extrinsic evidence and intrinsic evidence support the understanding that "internetwork operating system (IOS) command line interface (CLI) parser subsystem"—which

contains an adjective that Cisco protects by trademark—is the CLI interface designated by Cisco as IOS CLI, and that is used for configuring, monitoring, and maintaining Cisco devices.

4. "XML tag" in Claims 1-10

Arista's Proposal	Cisco's Proposal
"indicator that marks the start or end of an XML element and that is designated by a starting angle bracket, a corresponding closing angle bracket, and the content in between"	"one or a pair of XML indicators identifying data" "extensible markup language" or "XML" as construed above

"XML tag" is a technical term with a particular meaning in computer science. As defined in various technical dictionaries, a "tag" in a markup language is "generally a pair of angle brackets that contain one or more letters or numbers." (Ex. 7, Microsoft Dictionary at 511). "Usually one pair of angle brackets is placed before an element, and another pair is placed after, to indicate where the element begins or ends." (*Id*). The pair of angle brackets placed before the element is generally called the "start tag," and the pair of angle brackets after the element is generally called the "end tag." For example, the *Microsoft Computer Dictionary* shows a start tag, an XML element, and an end tag:

<LastName>Davalio</LastName>

(*Id.*) In this example, "<LastName>" is a start tag, "Davalio" is an XML element, and "</LastName>" is an end tag. Arista's proposed construction comports with this standard technical usage. Under Arista's construction, "<LastName>" and "</LastName>" are both tags because they include a starting angle bracket ("<"), a closing angle bracket (">"), and the content in between (e.g., "LastName"). But the XML element in between the tags ("Davalio") is not itself a tag. Under Cisco's construction, by contrast, it's not clear what qualifies as an XML tag. Cisco proposes that an XML tag is "one or a pair of XML indicators identifying data." What is an "indicator" that "identifies data"? Is an angle bracket itself an indicator? Is the whole string—including the XML element—an indicator? Cisco's construction provides no guidance, thus leaving a jury to guess as to what may fall under Cisco's broad definition.

As Cisco recognizes, the '886 Patent uses the term "XML tag" according to its ordinary

technical meaning. Cisco admits that the tag "<k_mpls_label>"—which appears in Table 1 of the 886 Patent—is an XML tag. (Cisco's Br. at 22:7-9). Arista agrees. But Cisco goes further and attempts to expand the scope of "XML tag" beyond its standard technical meaning. According to Cisco, the term may variously cover other formats because of the following passage in the specification:

> In step 370 of flowchart 300, in one embodiment, each XML tag has the following rule applied to it. If an XML keyword tag is a parameter node tag, the values inside the tag are extracted.

'886 Patent, 5:49-52. This disclosure does not make clear that the patentee adopted a non-standard definition of the term. *See Ancora*, 744 F.3d at 734. First, the passage does not even refer to "XML tag"; instead, it refers to "XML keyword tag." Second, it is not clear what is meant by "the values inside the tag." Is it the string in between the distinct start tag and end tag? Is it the string inside the two angle brackets? Is it some combination of the two? This hazy disclosure falls well short of the "clear and unmistakable disclaimer" required to alter the standard meaning of a claim term. *See Thorner v. Sony Computer Entm't Am. LLC*, 669 F.3d 1362, 1367 (Fed. Cir. 2012).

Because Arista's construction is consistent with the standard meaning of "XML tag," and because the patent's specification did not clearly alter or disclaim that standard meaning, the Court should adopt Arista's construction.

5. "XML parameter" in Claims 1-10

Arista's Proposal	Cisco's Proposal
"indicator within an XML tag that signals that the tag includes a CLI keyword"	Plain and ordinary meaning, except for "XML" as construed above

Due to space limitations, Arista submits its argument on "XML parameter" based on the evidence cited in the Joint Claim Construction Statement.

6. "parsing the output message to identify at least one CLI token" in Claims 1-10

Arista's Proposal	Cisco's Proposal
"breaking down the output message into its constituent character strings and determining	"analyzing the output message to extract at least one unit of CLI characters in a

that at least one such string corresponds to a CLI keyword or parameter"	sequence"

As with the parties' dispute concerning the construction of "command line interface (CLI) parser," the primary disagreement here is over what the word "parsing" means in the context of the patent. While Cisco and Arista agree that "parsing" needs construction, only Arista offers a construction that clarifies, rather than obscures, the meaning of the term. Cisco's construction replaces "parsing" with the even vaguer "analyzing." The term "parsing" in computer science refers to more than merely "analyzing"; it involve breaking down data into smaller pieces or chunks. (Ex. 6, Webster's at 275; Ex. 7, Microsoft Dictionary at 392).

The '886 Patent uses "parsing" in this sense when describing how the output message is parsed to identify at least one CLI token. The specification describes how the output message is traversed so that the "CLI statements are parsed into tokens," 886 patent, 6:24-27, which in turn correspond to CLI keywords or parameters, *id.* at 6:29-49. As demonstrated in Table 3 of the patent, the parsing of the message into tokens involves breaking down the message into keywords and parameters.

In short, Arista's construction "stays true to the claim language and most naturally aligns with the patent's description of the invention." *Phillips*, 415 F.3d at 1316.

7. "wherein the input command is configured in an extensible markup language (XML) format having a CLI syntax with CLI keywords sequenced according to configuration rules for CLI commands" in Claims 1-10

Arista's Proposal	**Cisco's Proposal**
"an extensible markup language (XML) format": "a format that complies with one of the versions of the XML standard published by the W3C organization" "keyword": "word describing an action or operation that the computer can recognize and execute" The entire phrase: "wherein the input command is written in an extensible markup language (XML) format, such that one or more CLI keywords, on the one hand, and the CLI	Plain and ordinary meaning, except for "extensible markup language (XML) format" as construed above

parameters, on the other, are contained within respective XML tags, and the sequence of the tags complies with the sequencing rules for keywords and parameters in the CLI grammar and syntax"	

Cisco argues that this technical claim phrase should be read according to its plain and ordinary meaning. But the "plain and ordinary meaning" of technical terms is only helpful when that meaning is "readily apparent even to lay [people]." *Phillips*, 415 F.3d at 1314. Here, how would a lay person know what a "CLI syntax with CLI keywords sequenced according to configuration rules for CLI commands" means to one of skill in the art in computer science? Some guidance is needed.

Arista's proposed construction provides that guidance in two ways. First, the construction includes a definition of "keyword," a term that is not defined in the specification. The definition is drawn from a contemporaneous technical dictionary definition. (Ex. 6, Webster's at 212).

Second, Arista proposes a construction of the entire phrase that clarifies its meaning in light of the description of the invention in the specification. Cisco complains that Arista's proposed construction attempts to "rewrite the claims so that it [*sic*] is limited to the embodiment shown in Fig. 3 and described at column 4:30–5:65 of the specification." (Cisco's Br. at 25). But the disclosure contained is *the only description of the claim phrase in the entire specification*, and Arista's construction captures the full breadth of that disclosure. The specification is "always highly relevant to the claim construction analysis. Usually it is dispositive; it is the single best guide to the meaning of a disputed term." *Phillips*, 415 F.3d at 1315*; see also Renishaw* at 1250 (Fed. Cir. 1998) ("[T]he interpretation to be given a [claim] term can only be determined and confirmed with a full understanding of what the inventors actually invented and intended to envelop with the claim.").

Arista's proposed construction, as well as the portion of the specification on which that construction is based, describes what "CLI syntax with CLI keywords sequenced according to configuration rules for CLI commands" actually means in the context of the patented invention. Accordingly, the Court should adopt Arista's proposed construction.

V. CONCLUSION

For the foregoing reasons, the Court should adopt Arista's proposed constructions.

Dated: December 7, 2015

KEKER & VAN NEST LLP

By: */s/ Robert A. Van Nest*
ROBERT A. VAN NEST
BRIAN L. FERRALL
DAVID SILBERT
AJAY KRISHNAN
MICHAEL S. KWUN

Attorneys for Defendant ARISTA NETWORKS, INC.